View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 50 (2015) 395 – 400

Procedia Computer Science

2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

Big Data Parallelism: Issues in different X-Information Paradigms

Koushik Mondal [a]*

[a]*System Manager, IIT Indore, Indore, MP, 452017,India*

Abstract

Emerging technologies are largely engaged in processing big data using different computational environments especially in different X-information systems such as Astronomy, Biology, Biomedicine, Business, Chemistry, Climate, Computer Science, Earth Science, Electronics, Energy, Environment, Finance, Health, Intelligence, Lifestyle, Market Engineering, Mechanics, Medicine, Pathology, Physics, Policy Making, Radar, Security, Social Issues, Wealth, Wellness and so on for different visual and graphical modelling. These frameworks of different scientific modelling will help Government, Industry, Research and different other communities for their decision-making and strategic planning. In this paper we will discuss about different X-informatics systems, trends of different emerging technologies, how big data processing will help in different decision making and different models available in the parallel paradigms and the probable way out to work with high dimensional data.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of scientific committee of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

Keywords: Big Data; Parallel Computing; High Dimensional Data; Scalable framework; Data Science; Machine Learning; Semi-stochastic; X-informatics;

1. Introduction

X-informatics is the study of the all-relevant fields in the light of information science to design a model, which fits the available data, with the help of necessary scientific tools. To understand the X-informatics, let us first discuss four different paradigms of scientific research viz. Theory, Experiment or Observation, Modelling or Simulation of

* Corresponding author. Tel.: +91-0731-2438714;.
E-mail address: gemkousk@gmail.com

1877-0509 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of scientific committee of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)
doi:10.1016/j.procs.2015.04.028

theory and Data driven experiments. In the present scenario, emerging technologies focusing on third and fourth paradigms of scientific research, as described in [1], depicted in Fig.1. The main aim of data science or big data modelling is to gain insights into data through computation, statistics and visualization.

According to Josh Blumenstock "A data scientist is one who knows more statistics than a computer scientist and more computer science than a statistician." Thus a data scientist is a combination of statistician, programmer, storyteller, coach and artist. Volume, Velocity, Variety, Veracity and Value --- these five "V" has often been used to describe the issues at hand in big data world. "Between the dawn of civilization and 2003, we only created five exabytes of information; now we're creating that amount every two days." – The comment of the Google team led Eric Schmidt ascertained the data generation volume and velocity are really high in the digital world. Needless to say that are of a wide variety of data range. Big data referred to the data driven analysis approach on these "big" amount of data in order to gain scientific insights, increase situational awareness, improve services and generate economic value. The new challenge that arises for use these x-informatics datasets in a machine learning framework is that it must be fit in some suitable optimization algorithm, in-memory or out-of-the-core capacity of the machine to handle large chunk of data to process and thus, huge computational requirements are there, to analyze those large x-informatics datasets.

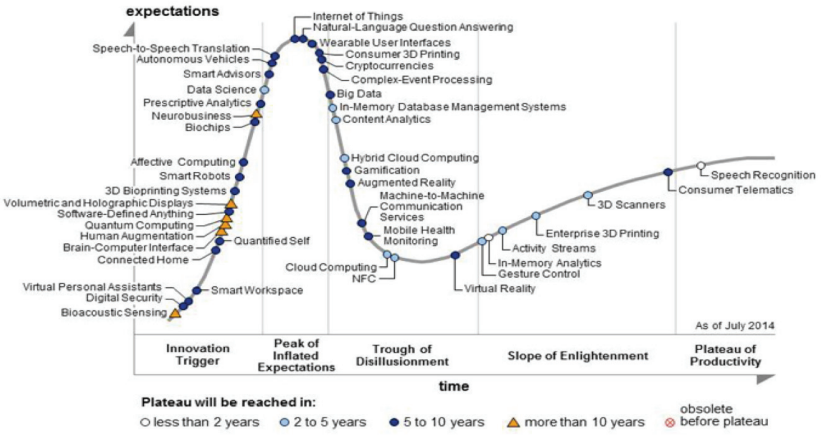

Fig. 1. Hype Cycle for Emerging Technologies Map 2014 [1]

2 Related Work

In recent years, many people engaged in developing frameworks for distributed computing applications for x-informatics datasets. Some of the most interesting and significant results have been the functional programming model used in MapReduce [2] with its associated frameworks such as Hadoop [3] and ApacheSpark [4] and vertex-centric models used in various graph based distributed frameworks such as Pregel [5], Hama [6], GraphLab's GraphChi [7], ApacheGiraph [8]. These frameworks are also help to build domain specific programming libraries such as Mahout [9] and Pegasus [10]. The Mahout library provides a set of machine learning algorithms and the Pegasus framework provides several Hadoop versions of different graph mining algorithms for massive graphs. A theoretical detail for the MapReduce framework is provided in [11]. The MapReduce paradigm's ability to efficiently express and iterative computation is limited, as it engaged itself in unnecessary data movement. To address this limitation, a modification to the MapReduce model, called Haloop [14], was proposed by Bu et.al. Based on Hadoop, Haloop is specialized for handling iterative problems with a modified task scheduler and the ability to cache frequently used data. Even after this there are several classes of algorithms, which do not fit the MapReduce framework. Bulk Synchronous Parallel framework (BSP) [15], developed by Valiant, aimed to provide a theoretical foundation for parallel algorithms that calculates communications and synchronization costs with computation for total model cost. Algorithms following the BSP framework contain three steps per iteration viz. computation, communication and synchronization. Hama project provides such BSP framework that runs on the top

of Hadoop distributed file system that attempts to improve data locality for matrix ad graph based algorithms.
The underlying mathematical models those are available in the machine learning domain for efficient parallel, iterative computation are described in [12] [13] [14]. The new challenges that driven the recent research is to model such kind of hypothesis functions which will able to predict unseen data accurately. The new trend demands criteria like --- large scale, stochasticity, non-linearity, parallelism, good generalization that have led to a burst of research activity. The broad two mathematical categories for designing large-scale parallel and iterative mathematical machine learning models are stochastic and batch. In general, batch method chooses a sufficiently large sample of training examples to define an objective function to minimize sum of the error terms using a standard gradient-based method.

3 Parallel Design Approach

We will discuss in three phases: First, how to design mathematical model for parallel environment; second, programming models available as of now and lastly different computational mechanisms.

3.1 Mathematical Model

The mathematical model for parallel environment is quiet different from sequential in setting. The stochastic gradient model is particularly difficult to parallelize [16], whereas batch methods are not. Thus we have to look for algorithms that will wok in both the environments efficiently. The complexity analysis does not consider newly developed methods that is neither purely stochastic nor purely batch. It is sometimes called 'semi-stochastic'. A typical implementation chooses a single training set (x_i, z_i) at random and updates the parameters of the model through following iteration. The method of this type gradually increases the size of the sample $|S_k|$, which is given by

$$w_{k+1} = w_k - \alpha_k \frac{1}{|S_k|} \sum_{i \in S_k} \nabla l(h(w_k; x_j), z_j) \quad \textbf{(1)}$$

Here α_K is a step-length parameter, l is the loss function that measures the derivatives between a prediction and the data, w_K is a random variable. This method emulates the stochastic gradient method at first but evolve toward batch gradient model. This approach enjoys work complexity as well as that of stochastic gradient model provided that $|S_k|$ increases geometrically with k.

3.2 Programming Models

We want to bring parallelism in a parallel programming environment either in terms of data parallelism or task parallelism with help of communication channels, concurrency, synchronization, locality/affinity management. These parallelisms offered us low level, control-centric, easy to implement environment. These parallelisms suffer from too many distinct notations for expressing parallelism, lots of user-managed details and locality. Over the time, different programming languages are developed to handle the different memory models like shared memory, distributed memory, in-memory, out of the core memory etc. efficiently. Most of the new generation programming languages is based on C, C++, Fortran and Java.

The following table will help us to understand hardware parallelism in a better way:

Table 1. Hardware Parallelism

Type of H/W Parallelism	Programming Model	Unit
Instruction level vectors /threads	Pragmas	Iteration
Intra-node/multicore	OpenMP/pthreads	Iteration/task
Inter-node	MPI	Executable
GPU/accelerator	CUDA/OpenCL/OpenAcc	SIMD function /task

3.3 Computational Mechanisms

Speed up in the performance of parallel computation is largely depends upon by how much of the job must be performed in sequentially. Amdahl's Law characterizes the speedup S that can be achieved by n processors collaborating on an application where p is the fraction of the job that can be executed in parallel and unit time for a single processor to complete the job. The main challenge to speed up in processing large x-informatics datasets is mainly depends upon the data movement between disk and main memory and parallel and non-parallel blocks in a program. To reduce the data movements between main memory and disk, creating and managing standard datasets for machine learning environment are crucial.

Some standard benchmark applications are PageRank on an unweighted graph [17], CoEM [21], Triangle counting, Spherical K-means clustering of sparse high dimensional vectors [18] and matrix factorization using stochastic gradient descent (SGD) for recommenders systems [19][20] are available and used extensively for big data experiments. For PageRank experiment, uk-2007-05 web graph, with 105 million vertices and 3.3 billion edges are considered. For CoEM 2 million vertices and 200 million edges are considered. We have considered a sparse document term matrix of newspaper article with 30 million rows and 83 thousand column containing 7.3 billion non-zeros for K-means experiments. For SGD experiments, we use dataset from NetFlix Prize, replicated 128 times to create an 8X16 block matrix, with 3.8 million rows 284 thousand columns and 12.8 billion non-zeros.

Let us now discuss MapReduce model that is greatly reduce the effort needed for large scale data processing using a class of software for processing big data. The MapReduce model is comprised of two phases: map and reduce. In the map phase, a list of key-value pairs are generated/gathered for a specific task and in the reduce phase, computation is performed on all of the key-value pairs from the map phase and the result is saved. This allows for task-parallel execution, where compute node can pick up map and reduce tasks to perform. Some popular and publicly available implementations of MapReduce are that of Hadoop, Pegasus and Pregel. We used Ubuntu 12.04 LTS for operating System and version 1.1.2 of Hadoop and OpenJDK IcedTea6 for creating the environment for experiment.

Another important aspect that we have to keep in mind during model design/selection is bulk synchronous and bulk asynchronous processing. In bulk synchronous machine learning paradigm, all the computations are completed in phases and then the messages are sent to the next phase. It is simpler to build, like MapReduce model. We need not have to worry about race conditions as barrier guarantees data consistency and make it simpler fault-tolerant. If we compare it with the matrix domain problem, it is similar to Jacobi iteration. Slower convergence for many machine learning problems is the main issue with synchronous bulk processing. In the other hand, asynchronous bulk processing is mainly used to address graph-parallel execution. It is harder to build as race conditions can happen all the time. We have to take care of fault tolerance issues and need to implement scheduler over vertices as vertices see latest information from neighbours. It is similar to Gauss-Seidel iteration. The convergence rate is quite high in asynchronous bulk processing.

4 Results and Discussions

We have performed experiments with different datasets in different programming environments, thus it will help us to differentiate different algorithm's performance. Table 2 represents the time required in PageRank applications. As we know contextual preferences determine a ranking of the data based on interest of the user but at the same time manually mentioning all the preferences seems unrealistic, thus we used automatic techniques for machine learning paradigm.

Table 2. Page Rank Performance

Algorithm	Time in min. when number of Nodes =1	Time in min. when number of Nodes =4
MPI	16.95	11.46
GraphChi	46.98	N/A
Hadoop	N/A	240.43

Table 3 describes the performance of Triangle Counting database that is having 40M users with 1.2B Edges. Different algorithms with higher level parallel abstraction try to simplify parallel abstraction by hiding challenges at different levels.

Table 3. Triangle Counting in Twitter Graph

Algorithm	No. of cores/Machines	Time required to process
Hadoop	1640 machines	423 mins.
GraphChi	1 Mac Pro with quad core	65 mins.
Graph Lab 2	64 machines 1024 cores	3 mins.

CoEM project database designed for named entity recognition task with 2 Million vertices and 200 Million edges. It first learns from a separate set of classifiers using labeled data. The instances with the highest confidence from the unlabeled data are then used to iteratively construct additional labeled training data in the subsequent phases. A combination of Co-Training and EM performs better than both Co-Training and EM. Thus combination of semi-supervised and active learning has shown a great promise in parallel environment. Table 4 illustrates the CoEM performance with different algorithms.

Table 4. CoEM Performance

Algorithm	No. of cores/Machines	Time required to process
Hadoop	96 cores	8 hrs.
Distributed GraphLab	32 EC2 machines	1.5 mins.

K-means usually incurs lower communication overhead than that of PageRank algorithm (broadcast/reduction vs. all-to-all). We can pose K-means algorithm as MapReduce where classify step of K-means will pose as Map phase and re-centre phase of K-means will represent by Reduce phase. Table 5 represents node-wise performance of K-means algorithm.

Table 5. K-means Performance

Node Details	Algorithm	#R=1/#T=1 Time in sec.	#R=4/#T=1 Time in sec.	#R=1/#T=4 Time in sec.
when no.of Nodes = 1	OpenMP	26.72	N/A	N/A
	MPI	44.86	N/A	54.25
when no.of Nodes = 4	MPI	19.23	N/A	15.23
	Hadoop	N/A	1198.25	N/A

#R represents no. of slave processes
#T represents no. of Threads

Stochastic Gradient Descent (SGD) is a common convex optimization procedure that is highly popular due to its simplicity and effectiveness on a broad range of problems. It is effective due to its tolerance to race conditions. Table 6 represents SGD's Performance on different algorithms. The MapReduce framework works optimally when the algorithm is embarrassingly parallel /data parallel and able to decomposed into large numbers of independent computations. The MapReduce abstraction fails when there exists computational dependencies in the data. For example, MapReduce can be used to extract features from a massive collection of images but unable to represent computation that depends on small overlapping subsets of images. Many machine learning algorithms like EM, SGD etc. iteratively refine some parameters, during both learning and inference, until some termination condition is achieved. MapReduce doesn't provide a mechanism to directly encode iterative computation. Thus it is obvious that a single computational framework is unable to express the all the large-scale features like dynamic, asynchronous, iterative computation etc. that is common to many machine learning algorithms.

Table 6. SGD Performance

Time in Sec.		Num. of Nodes =1		Num. of Nodes = 4	
Traversal Technique	***Algorithm***	***#R=1***	***#R=4***	***#R=1***	***#R=4***
Element wise random traversal	OpenMP	64.14	N/A	N/A	N/A
	MPI	> 2500	N/A	N/A	N/A
Row wise random traversal	OpenMP	29.24	N/A	N/A	N/A
	MPI	>2500	N/A	N/A	N/A
	GraphChi	61.28	N/A	N/A	N/A

#R represents no. of slave processes

Acknowledgment. The author is indebted to Prof. (Dr.) Raghunath Sahoo, Associate Professor, Department of Physics, IIT Indore for allowing the author to perform his computational work in the Tier-3 setup of HEP Lab.

References

1. Gartner's 2014 Hype Cycle for Emerging Technologies Maps the Journey to Digital Business, http://www.gartner.com/newsroom/id/2819918
2. J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of ACM 51, Vol.1, 2008, pp. 107–113.
3. Apache Hadoop. <http://hadoop.apache.org>.
4. M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica, Spark: cluster computing with working sets, in the Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 2010, pp. 10–14.
5. G. Malewicz, M.H. Austern, A.J. Bik, J.C. Dehnert, I. Horn, N. Leiser, G. Czajkowski, Pregel: a system for large-scale graph processing, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, ACM, 2010, pp. 135–146.
6. S. Seo, E.J. Yoon, J. Kim, S. Jin, J.-S. Kim, S. Maeng, Hama: an efficient matrix computation with the mapreduce framework, in IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2010, pp. 721–726.
7. Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, J.M. Hellerstein, Graphlab: a new parallel framework for machine learning, in: Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, California, 2010.
8. Apache Giraph, <http://giraph.apache.org>.
9. Apache Mahout, <http://mahout.apache.org/>.
10. U. Kang, C.E. Tsourakakis, C. Faloutsos, Pegasus: a peta-scale graph mining system implementation and observations, in the Ninth IEEE International Conference on Data Mining, ICDM'09, IEEE, 2009, pp. 229–238.
11. M.F. Pace, BSP vs MapReduce, Procedeeings of Computer Science, Vol. 9, 2012, pp. 246–255.
12. L. Bottou and O. Bousquet, The tradeoffs off of large scale learning, Advance Neural Information Process Systems, Vol. 20, 2008, pp. 161-168.
13. R.H. Byrd, G.M. Chin, J. Nocedal, Y.Wu, Sample size selection in optimization methods for machine learning, Mathematical Programming 134:1, 2012, pp. 127-155.
14. Y. Bu, B. Howe, M. Balazinska, M.D. Ernst, HaLoop: efficient iterative data processing on large clusters, Proceedings of VLDB Endowments, Vol. 3 (1–2), 2010), pp. 285–296.
15. L.G. Valiant, A bridging model for parallel computation, Communnications of ACM, Vol. 33 (8), 1990, pp. 103–111.
16. B.Recht, C.Re, S. Wright,F.Niu, Hogwild: A lock-free approach to parallelizing stochastic grapient descent, Advance Neural Informtion Processing System,s Vol. 24, 2011, pp. 693-701.
17. R. Rabenseifner, G. Hager, G. Jost, Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes, in 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, IEEE, 2009, pp. 427–436.
18. L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: bringing order to the web.
19. I.S. Dhillon, D.S. Modha, Concept decompositions for large sparse text data using clustering, Mach. learn. 42 (1–2) (2001) 143–175.
20. B. Recht, C. Re, Parallel stochastic gradient algorithms for large-scale matrix completion, Math. Prog. Comput. 5 (2) (2013) 201–226. <http://dx.doi.org/10.1007/s12532-013-0053-8>.
21. Nigam, K., & Ghani, R., Analyzing the effectiveness and applicability of co-training, in the Ninth International Conference on Information and Knowledge Management (CIKM-2000), 2000, pp. 86-93.